Checkout and Launch Control System

Real Time Control Application Software

Development Plan

84K00070-002

RTC Applications Software Development Plan

Approval :

Retha Hart
CLCS Program manager

Ralph Esposito
USA CLCS Program Manager

Concurrence :

Benjamin Bryant
CLCS Applications Software

Marty Winkel
USA CLCS Applications Software

Jeff Wheeler
CLCS User Liaison

Richard Ikerd
Application Product Team (USA)

Ken Hale
Application Product Team (NASA)

Norm Peters
Application Product Team (USA)

Prepared By : Real Time Control Applications Software
United Space Alliance
Kennedy Space Center, Florida

Supporting Document Note: Acronyms and definitions of many common CLCS terms may be found in the following documents: CLCS Acronyms 84K00240 and CLCS Project Glossary 84K00250.

REVISION HISTORY

REV	DESCRIPTION	DATE
Basic	Promoted per approval by signatories. Updated to standard format. ljp	5/12/98

LIST OF EFFECTIVE PAGES				
Dates of issue of change pages are:				
Page No.	**A or D***	**Issue or Change No.**	**CR No.**	**Effective Date****

Table of Contents

1.

1. IN

The Real Time Control (RTC) Application Software Development Plan (SDP) for CLCS must define the mechanisms to enable rapid development of software to satisfy CLCS milestones. This plan applies to both the CLCS initial project development effort for RTC Application Software as well as sustaining that software after initial development completion. RTC Application Software development is a major effort that must be flexible enough to capture existing requirements and to simultaneously incorporate both internal and external change drivers.

1.1 PURPOSE

This plan applies the concepts described in the overall CLCS System Software Development Plan (84K00070-100). The CLCS System Software SDP provides integration of the infrastructure (provided by system and application services) to support application software components that control and monitor the Shuttle, GSE and facility hardware. Although similar in concept, the RTC Applications Software Development Plan differs from the System Software SDP for the following reasons:

1. The RTC Application Software environment requires the active participation of the personnel with systems knowledge in all phases of the process (e.g., validation testing being performed by the user rather than an IV&V team) that necessitates a development/sustaining plan that is sufficiently different from the System Software world to warrant a separate document.

2. The System Software SDP relies upon a structured review board concept to ensure all project requirements are properly identified and captured. The realm of Shuttle/GSE knowledge is outside of the expertise of these review boards. The concepts presented in this document place the responsibility for proper requirements capture/definition and implementation on personnel who are qualified and knowledgeable for that function. The review panels for RTC Application Software perform an oversight and management function rather than a technical review function.

3. An internal applications team provides the day to day integration between application software and the system world. This method allows applications to support the big CLCS picture, while the teaming within applications supports the end user.

Almost as important as the plan is the background information that drives its creation. An understanding of the influences that mold the plan helps capture, for future users, the all important ‘why?’. As the ‘why’ changes over time, the process can be reevaluated. The following topics are part of the CLCS application software plan:

1. Application software goals
2. Productivity enhancement methods
3. General information
4. Applications development infrastructure
5. Application software development processing

The plan will explicitly define the differences between the RTC Application Software development and the post-development sustaining process.

1.2

SCOPE

This document applies to all RTC Application Software developed to support the CLCS project. It includes all reused and commercial-off-the-shelf (COTS) software except as noted.

1.3 REFERENCE DOCUMENTS

The following documents were used in the development of or are referenced in this SDP:

84K00050	CLCS Program Management Plan
84K00051	CLCS Project Plan
84K00052	CLCS Configuration Management Plan
84K00053	CLCS Systems Engineering Management Plan
84K00070-100	CLCS System Software Development Plan
84K00230	CLCS HCI Style Guide and Standards
84K01700	RTC Application Software Documentation Standard
84K01710	RTC Application Software Architecture Standard
84K01720	RTC Application Software Implementation Standard
84K01830-xxx	RTC Application Software Development Practices
84K07500-010	CLCS Programming Standard
DP-P-07-BASIC	RTC Application Software Development Flowchart

This plan is derived from the CLCS System Engineering Management Plan (SEMP), 84K00053, which defines the overall system engineering process for the CLCS project, and the System Software Development Plan, 84K00070-100. Selected information from these documents is included in this document to aid in understanding how the RTC Application Software development process fits into the overall system engineering process.

1.4 RTC APPLICATION SOFTWARE GOALS

The RTC Application Software SDP must establish a process that meets the following goals:

1. ***Safety*** - The process must ensure that personnel safety and Shuttle hardware is not compromised.

2. ***Cost*** - The process must be inherently designed for efficiency to reduce cost. There are several components of cost:
   - The overhead required to develop or make a change
   - The cost of actually performing the work
   - Potential monetary savings as a result of implementing a change
   - Long term product maintainability

3. ***Traceability*** - The process must provide a level of traceability from development/change driver through implementation.

4. ***Accountability*** - The process must provide a scheduling method to track due dates (constraints) and a method to capture the total cost of a development effort.

5. ***Repeatability*** - The process must be organized in such a way that it is deterministic and repeatable for all Application Software developed components.

6. ***Operational Knowledge 'Recapture'*** - Application Software must capture the operational 'whys'. When a system matures, 'why' a component or process exists is often lost (only the 'what' and 'how' remain). RTC Application Software development must maintain the 'why'. One purpose of this document is to 'recapture' the application process 'why' for future users.

1.5 PRODUCTIVITY ENHANCEMENT METHODS

In any project, change is inevitable. Limited funding restricts the amount of available resources, requiring development/change processing to be as efficient as possible to meet project goals. Several different methods will be used to enhance productivity:

1. A common 'style' for all of applications software will eliminate much of the specialization that occurs in applications software production. While aspects of a particular end item may be unique, the implementation 'style' will be consistent across systems. This consistency allows the creation of a 'pool' of software production personnel to assist in the application software sustaining effort. While it will be necessary to maintain a certain level of system specialization (to ensure consistency), a "pool" of Software Engineers will provide flexibility in managing both the development and sustaining efforts.

2. Reuse of common "building blocks", developed using a common style, will greatly enhance efficiency over the development of new products. Placing these "building blocks" in a common, well documented location further enhances reusability.

3. Teaming will be used to involve 'core' members in the entire process of RTC Application Software activities. Core members include personnel with expertise in system knowledge and software production. The traditional 'fence' between hardware and software functions does not exist with effective teaming.

4. Delineation of responsibility is a key factor in increasing productivity. The number of 'boards', 'reviews', and signature requirements will be the minimum required, without sacrificing safety or quality control. Signatures must mean something. Change management processing with large number of 'checkpoints' (signatures) typically suffer from an 'insulating effect'. When the number of 'required' signatures increase, insulation occurs, the assumption being 'someone else' will catch any problems.

5. Rapid prototyping will be used to establish the 'best' path that enables RTC Application Software to reach its milestones. Prototyping allows for the development of streamlined and efficient processes before initiating all product teams. Prototyping is best performed by a limited number of 'pathfinder' projects. The products developed during the prototyping efforts should be treated as potential throw-away products.

6. Communication of both success and problem areas encountered during the process are integral to productivity. The focus, historically, has been isolating and correcting problem areas. Applying success stories (discovered from within RTC Application Software and from the external world) throughout the development process has the potential to increase overall productivity significantly.

1.

2. GE

This section describes the basic elements that affect the development and sustaining of RTC Application Software. The applications workforce exists to implement the requirements that are driven by varying change sources. The level of effort necessary to perform this implementation is dependent upon the type of development/change effort and the software component criticality.

2.1 APPLICATION SOFTWARE PARTICIPANTS

Conceptually, there are four 'classes' of personnel who support RTC Application Software development:

- Management
- Integration
- Implementation (Development/Sustaining)
- User

Each 'class' of personnel has two primary elements associated with it:

- *Systems Knowledge* - provides the why and what of the software
- *Software Implementation* - provides the how of the software

2.2 CHANGE SOURCES

There are two primary sources of change that affect RTC Application Software development: external drivers and internal drivers. Change is an asynchronous process, with the type and number of drivers appearing at random. The sources are:

1. External which includes Shuttle design center changes, KSC generated (ground) system modifications and CLCS system software changes. These changes typically are mandatory, requiring an implementation by a given milestone. Often the change provides a funding source. These changes are the most 'random' since the external source is driven by influences outside RTC Application Software.

2. Internal change sources originate from within the applications community. These include operational enhancements to existing software, new software components that are required as the result of daily operations and changes required to correct software discrepancies. Most of these changes do not have mandatory milestones (excluding some discrepancy processing), but do have a level of relative priority. Because these changes occur from within, they offer the potential of planning implementation.

During the CLCS development process, external change drivers will need to be incorporated into CLCS application deliveries.

2.3 CHANGE EFFORT

The amount of effort to implement a change can be grouped into two categories: minor and major. Change sources (both internal and external) can impose different levels of effort. Adequate personnel resources must be available to handle concurrent levels of effort. The RTC Application Software process will make a distinction between minor and major change processing.

1. A major change requires more than a few days to implement. Often the change driver has an impact on several systems. These changes require a more formal integration effort to ensure adequate scheduling and availability of resources. A change is considered major if the total change implementation effort estimate exceeds thirty (30) man-hours (for all activities).

2. A minor change effort is of limited scope, can be assessed quickly and can be implemented in a minimum of time. In the classical change management system, the cost of performing a minor change is often dwarfed by the overhead required to support the change. A change is considered minor if the total manpower estimate is less than or equal to thirty (30) man-hours (for all activities).

2.4 SOFTWARE COMPONENT CRITICALITY

Software component criticality is based on how the application software component is used. Application component criticality may be different from the criticality assigned to the hardware component. The criticality classification is used to determine the level and type of quality oversight required during the different phases of development. There are three levels of 'criticality' in the RTC Application Software environment. A component 'bubbles' to the highest level of criticality applicable. Software Classification Practice 84K01730-102 defines the process for classifying software modules.

1. Critical components are the "front line" items whose operation must be ensured to allow a successful commitment of the vehicle to spaceflight. As a minimum, this includes the Launch Sequencing/Launch Commit function, and the propellant loading function. The criticality of these components requires a level of quality support above the baseline.

2. Sensitive components encompass the control and monitor of items that, if not operated properly, could result in personnel injury or hardware damage. This class of items includes "reactive" responses to off-nominal conditions and the control of high energy systems (e.g., cryogencis, hydraulics, power units, etc.). The sensitivity of these components requires a minimum level of independent quality support to ensure proper validation of requirements.

3. Operational support components make up a large portion of the RTC Application Software products and are classified as non-critical. These are used for system monitor support and the control and monitoring of non-hazardous systems and the monitoring of systems where "reactive control" has been delegated to 'external systems' (e.g., a smart HIM, on-board sequence, etc.). An operational support component would not have an associated time critical safing function. The benign nature of these components requires less quality oversight.

1.

3. AP

This section details the key structures that have been established for RTC Application Software development. When viewed as a whole, these items establish the infrastructure that enables applications to obtain its goals. The infrastructure enables efficient change processing during both the development and the sustaining process.

The RTC Application Software development process provides the following key components:

1. Streamlined functional structure
2. Resource pool
3. Assignment of responsibility to an appropriate level
4. Automated component testing
5. Iterative testing
6. Built-in quality philosophy
7. Value added metric gathering
8. Meaningful product documentation

3.1 STREAMLINED FUNCTIONAL STRUCTURE

There are three levels of functional groups that directly participate in RTC Application Software production:

3.1.1 Application Software Project Management

Application Software Project Management provides an interface to the CLCS project management team during the development phase. Applications management is chaired by NASA with support from the SFOC contractor. Both the NASA and SFOC Applications Software project management team provide overall guidance, resource acquisition (hardware, software and personnel), provisions for training, and project level issue resolution / status. Application Software project management charters the Application Software Production Team (APT) to provide technical leadership. As the project moves into the sustaining mode of operations, the Management interface will transition from CLCS to SFOC.

3.1.2 Application Software Product Team

The APT provides the working level technical direction and oversight of the production of RTC Application Software. The APT is three individuals representing Systems Knowledge (Shuttle Engineering) and Software Production (Applications Software). The APT functions as a single ‘unit’ with each member having knowledge in all the processes required for applications development. When an area of specialization is required, the APT member with the specific expertise takes the lead. The APT charters Integrated Product Teams (IPTs) to provide specific application software products. The APT interfaces with Systems Software to resolve Applications/System Software issues. The allocation of resources and resolution of inter-IPT issues are also APT functions. The APT provides common standards, practices, and development tools to all IPT’s.

3.1.3

Analysis and Design Group

The Analysis and Design Group (ADG) is a collection of people from both the Systems Knowledge and Software Production disciplines. Their purpose is to translate the User Requirements provided by the individual Shuttle Engineering groups into a Functional Requirements Document (FRD). The members of the ADG are not assigned to an IPT. While members of the ADG interact with IPTs associated with the User Requirements they are translating, the ADG is independent from the IPTs.

During this effort, the ADG is responsible for:

- Coordinating with the Shuttle Engineering groups to ensure correct interpretation of the User Requirements.
- Eliminating duplication of requirements by identifying those requirements that are common between systems. This paves the way for infusing software reuse into the development effort.
- Ensuring all requirements are testable and do not contain any implementation specific details

3.1.4 Integrated Product Teams

Each individual RTC Application Software IPT is a collection of people with a diverse set of skills that are tasked to deliver a CLCS application software capability. The scope of an IPT may encompass multiple, related systems. IPTs provide an oversight function to the software development effort to ensure User Requirements are satisfied and that the end users become familiar with the software on an increment basis. Each IPT is comprised of :

1. Systems Knowledge personnel (e.g., Hardware Engineering, System Engineering, Test Engineering)
2. Software Production personnel (e.g. Software Engineers, Software Developers)
3. Simulation Support personnel
4. Others as required

Each IPT is comprised of smaller sub-teams that perform the work assigned to them. Figure 3-1 provides an illustration of an IPT structure. The IPT is the mechanism for coordinating the activities of these sub-teams.

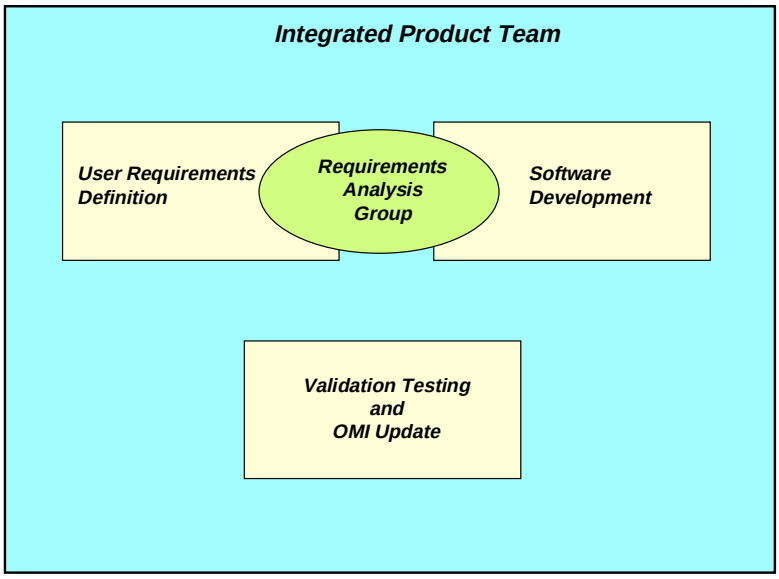

Figure 3-1 : Integrated Product Team Structure

3.1.4.1 Requirements Definition Team

The responsibility of this sub-team is to capture the user requirements in the Functional Requirements Document. This team is comprised of representatives from the Shuttle Engineering discipline, Software Engineering and the ADG.

3.1.4.2 Software Development Team

Upon completion of the Functional Requirements, the Software Development Team transforms the FRD requirements into a software implementation. This team is comprised of individuals knowledgeable on software development practices, languages and tools. This team is responsible for:

- Developing design documentation, software code and for performing unit and integrated debug testing
- Maximizing the reuse of software components to reduce the duplication of code and testing
- Providing coordination with the IPT on technical issue resolution and schedule progress
- Providing any simulation math model upgrades that are a result of new requirements

3.1.4.3 Software Validation Team

The Software Validation Team is responsible for the final validation testing of the software developed for the IPT and for the update of existing Operations and Maintenance Instructions (OMIs) to incorporate use of the new software. This team is comprised primarily of Shuttle Engineering personnel with assistance from the Software Development Team as required. This team is responsible for:

- Development of all validation test procedures. These procedures will test the software against the Functional Requirements Document.
- Conducting all validation testing.
- Update of all OMIs affected by the software changes.

3.2 RESOURCE POOL

The APT is responsible for the allocation (by number) of Software Production and Systems Knowledge personnel to perform applications activities (acquisition of identified resources is performed by management). In the sustaining environment, each IPT will be allocated a baselined manpower equivalent, typically systems knowledge and software production individuals (on a potentially part time basis). This allocation is designed to provide each IPT with the capability to perform minor change processing quickly and efficiently. For major changes, the IPT will identify resource requirements to the APT. The APT is responsible for prioritization of tasks when the resource pool approached being empty.

3.3 ASSIGNMENT OF RESPONSIBILITY TO AN APPROPRIATE LEVEL

Providing a solid infrastructure which includes a charter (definition of responsibility) and a common approach, the RTC Application Software process assigns the responsibility for producing products at the lowest level possible. The key in the assignment is establishing open lines of communication between the responsible functions. Problems will be communicated before they become issues and non-issues need not be continually 'statused' by higher levels. The bulk of the applications production occurs at the IPT level and this is where the largest portion of responsibility resides. Figure 3-1 illustrates the RTC Application Software Functional Structure.

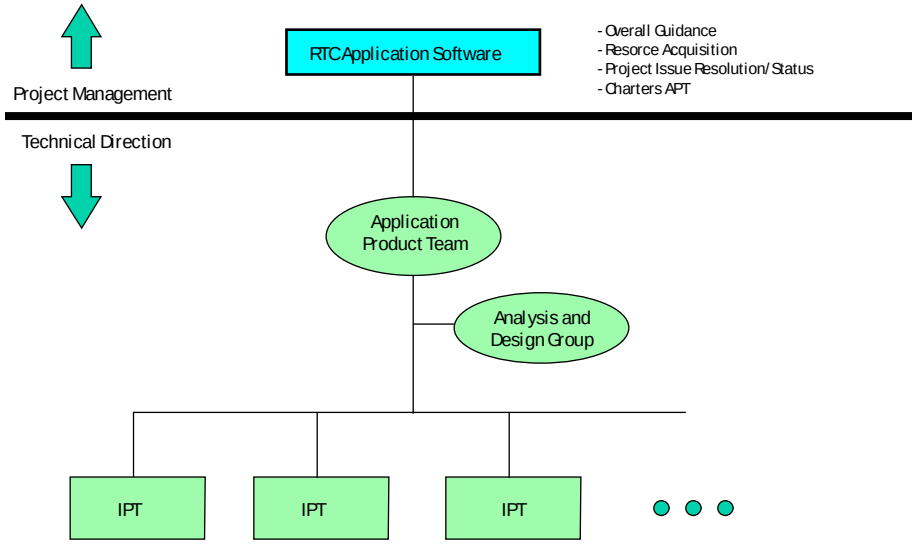

Figure 3-1 RTC Application Software Functional Structure

1. Project Management has the following responsibilities:
   - This oversight functionality is comprised of:
     - NASA Application Software Division Chief
     - SFOC RTC Application Software Manager
   - Overall project guidance
     - In the development phase, provide a common focal point for all non-technical project level issues (e.g., delivery schedules)
     - In the sustaining phase, provide a common focal point for ensuring the production activities fit within the scope of the Shuttle program guidelines.
   - Budget and Resource Management

   Acquire the resources necessary to develop and sustain RTC Application Software through determination and management of the required labor and non-labor budget. This commonly includes:
     - Acquisition of personnel
     - Acquisition of any hardware required to develop RTC Application Software
     - Acquisition of COTS Software packages required to develop RTC Application Software
     - Identification and acquisition of training for development and user personnel

2. The APT has the following responsibilities:
   - The APT is comprised of:
     - Systems Knowledge Representative
     - Software Production Representative
     - NASA Application Software Division Representative
   - Charters each application IPT and establishes core participants
     - Uses the master RTC Application Software delivery schedule as input
     - Identifies a “focal point” for each sub-team within the IPT
   - Performs the integration function among IPTs and between RTC Application Software and the CLCS project (on a technical level)
     - Provides an oversight function to ensure commonality (in appropriate areas) across all IPTs
     - Allocates resources to meet deliveries/schedules
     - Resolves inter-IPT issues
     - Coordinates resolution of CLCS issues pertaining to RTC Application Software
     - Coordinates with System Software services for deliveries and their contents
   - Provides common processes, tools, directions
     - Responsible for the development and maintenance of this document
     - Responsible for the development and maintenance of all standard practices and guidelines associated with Applications Software
     - Responsible for defining and evolving the Applications Software design standard
     - Provides input to the CLCS programming standards and guidelines
   - Chairs the Application Change Review Panel (ACRP) and the Requirements Review Panel (RRP)
   - Provides development status to Delivery Manager
   - Presents IPT concepts to CLCS Concept Design Panel

3. The Analysis and Design Group has the following responsibilities:
   - The ADG is comprised of:
     - Systems Knowledge Representatives
     - Software Production Representatives
   - Definition of the Functional Requirements Document format
   - Development of a common FRD that contains generic requirements that can span multiple systems. These “common” requirements can then be referenced in a system’s FRD rather than duplicating them.
   - Participation in the development of system FRDs to help ensure consistency between systems
   - Identification of test plans for all requirements. These are not a step-by-step procedure, but are outlines that describe the functionality that needs to be tested.

4. IPTs have the following responsibilities:
   - Each IPT is comprised of:
     - System Knowledge personnel
     - Software Production personnel
     - Simulation personnel
   - Refining the IPT charter
   - Developing a detailed schedule
   - Identifying resource requirements
   - Following processes and standards established for application software
   - Developing applications software products
     - Software
     - Documentation
     - Procedures

3.4 AUTOMATED COMPONENT TESTING

RTC Application Software will make every effort to use automated COTS test tools. Automated routines (developed in a test control language) will be stored in a common library. The routines provide repeatable ‘difference based’ methods to validate component software. Levels of test routines can be created to test larger control ‘parts’. For example: software production personnel will use scripts to test ‘atomic’ components, higher level scripts will test assemblies, and even higher scripts will test modules. Test scripts will be developed to test all paths available. This hierarchy of automated test scripts enables all software involved in a change to be fully tested in a minimum of time; unlike the CCMS verification which typically only test the differences because of the extensive manual effort that is involved.

3.5 ITERATIVE TESTING

Requirements verification testing at each phase is a key part of the CLCS design philosophy. The first two phases are performed in the development environment with subsequent phases moving to an operational environment. Throughout the development process, informal testing will be performed to make sure the desired operability is provided. This informal testing ensures problem areas are addressed early in the development cycle. IPTs will perform the following types of testing:

1. Detailed End Item Component (object) level testing will use automated routines (where applicable) to test every path supported by the component.

2. Detailed module testing will validate the functionality of assembled components.

3. Interface testing of the assembled modules will be performed (again using test scripts). The interface testing begins to test operational control concepts. The CLCS transition strategy breaks this phase into distinct pieces: Integrated Application and Systems Integration. When testing the Integrated Applications portion, much of the Systems portion is exercised.

4. Validation tests are performed to verify application functionality. The scope of validation will change depending on the level of effort required to implement the change and on the software component's criticality classification. For example, when the internals of a component change (without affecting the external interfaces), the component will be thoroughly tested and a minimal interface test will be performed. Problems detected during validation will be documented and tracked outside the formal PRACA system (using the Configuration Management tool). Only problems not corrected prior to the software being released as "ready to use" will require formal PRACA documentation. When validation has been completed, the applications component is considered 'ready for use'.

5. The final level of testing is User Acceptance. User acceptance verifies that the applications product not only implements the requirements, but can be operated by the variety of system users. User acceptance includes stand-alone, cluster and integrated simulation practice via a simulation model. Problems detected in the User Acceptance phase require formal documentation and remedial action. In the sustaining environment, user acceptance is implied during the validation phase for minor changes and, depending on the impact, may require formal acceptance for a significant major change.

3.6 BUILT-IN QUALITY PHILOSOPHY

The two previous sections set the stage for built-in quality. Each of the five phases of testing uses informal and formal processes that help ensure the delivered product meets requirements. A quality function is provided as the formal check in the system. Details on the quality role are located in the Application Development Process section.

In CLCS, the Application Software quality assurance function has been redefined. Software quality transitions from the CCMS role of quality checker to the CLCS role of quality verification. This is not meant to down play the importance of quality in the CCMS environment. It is a re-focus to make use of unique skills and to remove redundancy. This will closely align the quality processing function of Product Assurance Engineering (PAE) with that of Hardware Engineering (QE). The RTC Application Software process utilizes quality assurance to:

1. Participate in the validation of critical and sensitive components
2. Participate in the validation of 'common' components
3. Identify regression test requirements
4. Verify test requirements (including regression testing) have been satisfied

5. Perform random audits of application test procedures to verify test processing activities have been followed correctly

Common applications (those used by multiple systems) require a third party (quality) test perspective. Quality support ensures the big picture is covered, since system engineers may only be interested in their specific application. It also provides the proper level of oversight to enable a component to be used in critical and sensitive software without requiring additional testing of the component. Critical application component validation will also have quality oversight because of the requirements to ensure successful spaceflight.

Systems knowledge personnel, along with software production personnel, will follow standard procedures to validate system specific applications. This implementation mirrors the Responsible Organization Representative (ROR) implementation used during shuttle Operations and Maintenance Instruction (OMI) testing. The ROR is responsible in ensuring the test procedure is performed successfully. Quality will no longer be required to participate in every application component validation.

Regression test identification moves under the quality umbrella. The 'big picture' role places quality in the best position to ensure the appropriate level of regression testing is performed. The mechanics of regression testing are still under refinement.

3.7 VALUE ADDED METRIC GATHERING

Metrics are an important tool in understanding strengths and systemic weaknesses in the applications cycle. Uncontrolled metric gathering tends to impede efficiency (the process spends too much time just acquiring data). Gathering the correct type of data, concisely, and applying the data in a meaningful manner adds value. Before a metric is instituted, a rationale will be developed that defines why the metric exists, what the metric is to measure (including weights) and how the metric is planned to be utilized. As the Applications process evolves, metric rationale will be reviewed and refined striving to obtain value. Throughout the metric process, one-time 'perturbations' will be captured, noted, but not factored into the overall picture. Perturbations often skew meaningful data masking the true reason behind the metric. Document 84K01730-103 describes the RTC Application Software metric program.

3.8 MEANINGFUL PRODUCT DOCUMENTATION

The RTC Application Software documentation structure has been created to aid in all facets of change processing. Two products are part of each CLCS CSCI: Functional Requirements and Software Design Specifications.

1. A Functional Requirements Document (FRD) will be created for each Shuttle, GSE and Facility system (reference Appendix B for a list of Application Software CSCIs). The FRD is developed using a COTS tool (DOORS) to aid in organizing requirements and enabling linking between requirements modules. In addition to the subsystem FRD's, a common FRD will be created to capture and store all reused items. Subsystem FRD's are implemented at the hardware functionality level (independent of the organizational level).

2. There will be two levels of the Software Design Specification (SDS). Each level addresses a different development need:
   - The Overview Design Specification (ODS) describes the RTC Application Software CSCI and all of its elements. This document will be developed prior to the software production phase and will provide the foundation for detailed design and coding.
     - Each element in the CSCI will be identified along with a description of its function and a high level design description.
     - Interfaces between the CSCI's components will be defined.
     - Interfaces between the CSCI and external elements will be defined.
     - A cross reference between the FRD and the ODS.

This document, while providing a good summary of each program in the Application Software CSCI, requires less frequent updating than the CCMS equivalent.

   - The Detailed Design Specification (DDS) will be generated from in-line comments in the source code using an auto-documentation tool. This documentation will be generated for web-based viewing and will not be printed. This is the information on the implementation techniques and reasoning that belongs primarily in the developer's world.

The redlining of application software documentation "for approval" will be reduced significantly. Configuration mechanisms will be implemented that allow direct input of redlines into the document. In CCMS, the maintenance effort of redlined original is significant, documents are often redlined years before the software implementation is scheduled. In CLCS, the concept is to submit (or effect) the documentation change prior to the release or acceptance of the product. RTC Application Software CSCIs are still required to produce documentation, it is now "formally" produced later in the development cycle.

The RTC Application Software documentation should be viewed in the context in which it is intended: it reflects the final implementation of a requirement (unlike other Shuttle program documents, which are used as change drivers). Typically, RTC Application Software documentation is used internally (although there may be external reviewers). For example: Space Shuttle Operational Increment (OI) changes are external forces that 'drive' the development effort. The FRD and Specifications are the KSC implementation of the external driver (FRD and Specifications have no influence on the OI change).

The RTC Application Software process requires significant 'up-front' effort, breaking the tendency to deliver a product, then fix it. The Applications software effort revolves around value added repeatable processes and procedures. The object is to do it once and to do it right the first time.

1.

4. RT

This section describes the development processes used by RTC Application Software. Applications software development in CLCS is fundamentally structured for efficiency. The core processing flow applies to all development efforts (including change drivers), it is, however, customized for different change types. The complete process is used for major development/internal changes and is streamlined for minor “operational” changes. The use of distinct paths obtains a much needed balance between productivity and process. NOTE : Since this section describes the process for both new development and sustaining work in response to a change driver (sustaining engineering), the term Statement of Work (SOW) will be used to denote either type of request for software development.

There are four major processes used in RTC Application Software development and sustaining. Figure 4-1 illustrates their relationship. DP-P-07-BASIC RTC Application Software Flowchart provides a detailed flowchart description of the development process.

1. Change Screen Panel Processing
2. IPT Statement of Work Assessment
3. IPT Statement of Work Implementation
4. Validation and User Acceptance

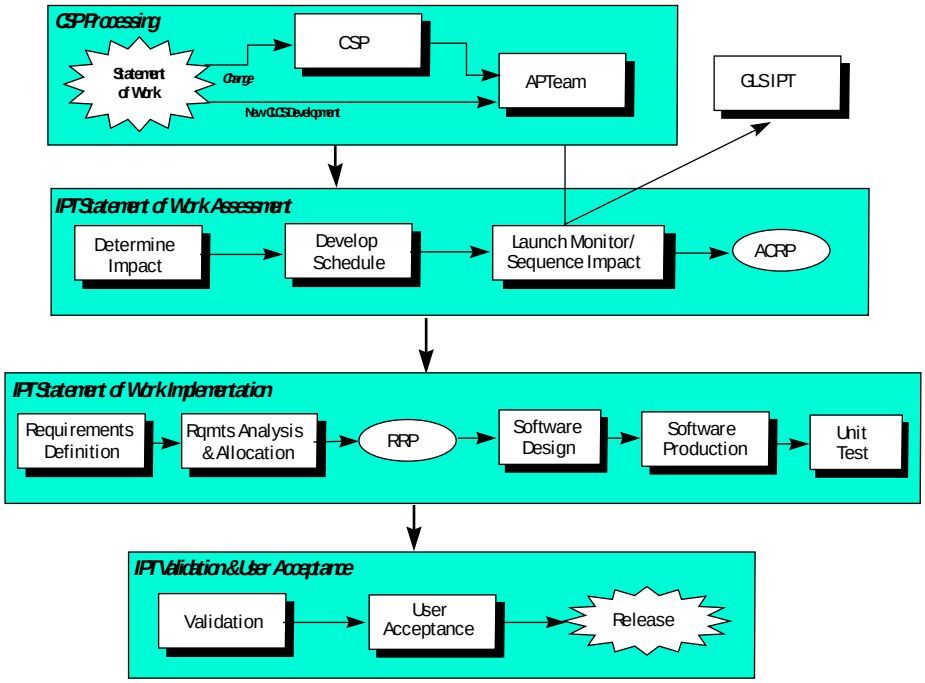

Figure 4-1 RTC Application Software Development Process Overview

4.1

CHANGE SCREENING PANEL PROCESSING

The Change Screening Panel (CSP) is responsible for determining which RTC Application Software CSCIs may have an impact to a change driver. The functional structure of Applications requirements and specifications will allow the CSP to query WEB based documentation starting from 'common use items' then branching to system specific items. The CSP will route the change driver to the IPT responsible for the affected CSCIs, allowing enough time for a response. In cases when the IPT cannot be identified or when the change driver is multi-system, the package will be routed to the APT for disposition. The APT is responsible for the disposition of integrated change drivers to (1) the appropriate IPT or (2) forming an IPT to work the integrated issue. All change driver implementation is performed at the IPT level.

During the CLCS development phase, the CSP will use the existing CCMS process to determine which Application Software Working Teams ASWT(s) will assess the change driver. In addition to the required CCMS routing, the change driver will be forwarded to the APT for information only. The APT will distribute the package to the appropriate IPT. The IPT is responsible for incorporating any change driver into RTC Application Software products when that change driver has an effectivity before the IPT's target application delivery date. When a change driver has an effectivity after the IPT delivery target, the RTC Application Software change processing model will be used to provide traceability, accountability, and cost tracking.

4.2 IPT STATEMENT OF WORK ASSESSMENT

After receiving a SOW, the IPT will assess the package for impacts to their associated CSCIs. To develop this assessment, the IPT will determine and document the following:

- Scope (i.e., extent of the effort)
- The criticality of the SOW using the criticality of the software components involved in the package. The SOW criticality is based on the highest level of the software criticality affected.
- Manpower and resource requirements (ROM estimate). It is not necessary to identify each component affected at this point of the process.
- Implementation schedule that satisfies the SOW milestone
- Interfaces with other IPTs (e.g., GLS IPT lead when the SOW affects launch monitoring/sequencing functions)

There are two levels of approval processes for the SOW based on the SOW source and the level of effort required to implement the SOW requirements.

4.2.1

Major Statement of Work Assessment

For major SOW driven by an external source (from Level 3 or other design centers), the IPT will perform the assessment and forward the documentation (signed by the IPT core members) to the APT and Configuration Management. Implementation can proceed without further approval (since the SOW has already been approved by a higher authority).

For all other major SOWs, the IPT will perform the assessment and present it, along with rationale for the SOW, to the Application Software Change Review Panel (ACRP). The ACRP will determine if the SOW meets approval criteria. Implementation cannot proceed without ACRP approval.

4.2.2 Minor Statement of Work Assessment

For both externally and internally driven minor SOWs, the IPT will perform the assessment and forward the documentation (signed by the IPT core members) to the APT and Configuration Management. Implementation can proceed without further approval. A follow-up report to the APT will be required, documenting the actual time spent on the SOW. If, during the SOW implementation, it is determined that the manpower expenditure will be greater than thirty man-hours, the IPT must re-assess the SOW and obtain APT approval.

4.3 REVIEW PANELS

Two milestone reviews are implemented to ensure application goals are addressed. The ACRP is the first process in the SOW implementation which provides initial insight into the SOW impact. A Requirements Review Panel (RRP) is the IPT's presentation to the APT that finalizes the impact assessments (both manpower and resources). The RRP occurs after the Requirements Capture and Analysis phases are complete before actual software development starts. This is not another approval point, but rather a means for the APT to ensure the SOW is on-schedule and that sufficient resources are available for assignment to the IPT to meet their schedule.

4.3.1 Application Change Review Panel

The Applications Change Review Panel is the first step of the development process. This is where the concept of a SOW is presented, with rationale, scope and estimated implementation costs, for approval to proceed. This panel requires only a minimum of effort be expended before presentation to the panel to potentially avoid unnecessary work. The panel's primary responsibility is to ensure the SOW meets the requirements of the program, that the SOW rationale meets acceptable criteria and that the estimated cost can be absorbed by the available budget and manpower resources. The board is not responsible for ensuring the technical content of the SOW is correct. This is the responsibility of the presenting IPT. After approval by the ACRP, no further approvals are required to proceed.

During the CLCS development phase the ACRP function is performed by the CLCS Systems Engineering and Integration (SE&I) Team. The CLCS Concept Design Panel (CDP) is the development version of the ACRP. The CDP is a formal presentation of delivery thread requirements and initial assessment to the CLCS Design Panel. The APT, in coordination with the IPT Lead assigned to the delivery thread, is responsible for coordinating, developing and presenting the IPT concept to the CDP. This presentation validates that all high-level information is in place to understand and implement the thread requirements identified in the CLCS Delivery Document.

This is the APT's "tag-up" that the CSCIs to be developed falls within the overall scope of the CLCS project.

The ACRP (or CDP) :

- Ensures developers understand the scope of work
- Confirms developers understand the delivery requirements and the IPT charter (applicable to the CLCS development phase only)
- Coordinates issues and relationships with other threads/capabilities
- Confirms identification of impacted CSCIs and dependencies on other CSCIs
- Coordinates risk issues and risk management efforts

Required products for the panel:

- Presentation material developed using the CLCS Concept Design Panel template (reference 84K00070-100 CLCS System Software Development Plan Appendix A)
- Preliminary implementation schedule
- Preliminary assessment (list) of affected CSCIs
- Preliminary assessment of required resources (ROM estimate of effort and other required resources)

4.3.2 Requirements Review Panel

The Requirements Review Panel (RRP) is the primary review given to an IPT's activities (that has project level visibility) prior to entering the implementation phase of development. It is not a technical review of the functional requirements since the associate Shuttle Engineers have the best knowledge in that area. Instead, it is a review of the progress made to date and of the implementation phase planning.

The goals and objectives of the RRP are:

- Ensure required products have been completed
- Review of the software element allocation
- Review and approval of the software classification assignments
- Review and approval of the implementation schedule and resource allocation request
- Verify that an internal technical review was conducted
- Review of the CSCI identified external interfaces to ensure compatibility across all Test Set CSCIs.
- Discussion of any issues the team needs addressed by the APTeam

Prior to the RRP, the IPT (or the CSCI leads as a minimum) shall have an informal requirements review with the affected Shuttle Engineering subsystems. This meeting should be a technical review of the requirements to ensure they capture the appropriate operational knowledge.

4.3.2.1 Membership

The RRP membership is comprised of representatives from Shuttle Engineering and RTC Application Software Engineering.

- APTeam USA Shuttle Engineering Representative (co-chair)
- APTeam USA Software Engineering Representative (co-chair)
- APTeam NASA Software Engineering Representative (co-chair)
- NASA CLCS User Liaison Representative
- USA CLCS User Liaison Representative

The RRP is an open forum and any interested individuals may attend as desired. Meeting minutes, action items and results will be posted to the RTC Application Software web site.

4.3.2.2 Required Products

Prior to scheduling the RRP, the IPT shall have the following documents available for review. These products should be available for on-line viewing (hard copies are not required).

1. Functional Requirements Document (FRD) for each CSCI associated with the IPT (input into DOORS using the approved FRD template)

2. Preliminary Overview Design Specification (ODS) for each CSCI associated with the IPT (input into DOORS using the approved ODS template)

3. Initial Requirements Traceability Matrix (generated from DOORS after requirements have been allocated to software elements)

4. Refined, from the Concept Design Panel baseline, planning schedule (formatted per 84K01730-107 Development Schedule Practice)

5. Detailed Implementation Schedule (to the software module/component level) formatted per 84K01730-107 Development Schedule Practice

6. Resource requirements statement identifying the number and type of personnel required for implementation and testing, including the time-frames required

4.3.2.3 RRP Process

The following paragraphs describe the RRP process.

1. During the initial IPT Statement of Work assessment, the IPT will estimate a date for the RRP. This date will be added to the RTC Application Software master schedule to provide insight into IPT activities.

2. Since RRP meetings are not held on a regular basis, it is incumbent upon each IPT to coordinate a meeting time with the RRP membership whenever the IPT is ready for the review. This coordination should be done approximately one week in advance of the requested date.

3. At the RRP, the IPT lead will present the following information. The format for the presentation is at the discretion of the IPT lead, but the specified information must be covered.
    - An FRD overview
        - Describing the capabilities specified by the requirements
        - Identification of any subsequence FRD activities
    - An ODS overview
        - Describing the mapping of the requirements into the RTC Application Software architecture
        - Identification of Critical and Sensitive software modules
    - Overview of the implementation schedule and resource requirements
    - Overview of the validation approach (e.g., facilities required, resources, special tests)
    - Discussion of OMI update planning

4. Upon acceptance of the RRP presentation:
    - All FRDs will be placed under informal Configuration Management control This establishes the requirements baseline for the effort which can then be modified per the SDP process.
    - All ODSs will be placed under informal Configuration Management control. This establishes the design baseline for the effort which can then be modified per the SDP process.
    - The implementation schedule and resource requirements statement will be added to the RTC Application Software master schedule for tracking.

Note: Placing the FRD and ODS under informal Configuration Management control provides the ability to indicate to the project that a major milestone has been accomplished and also provides the ability to gather requirements and design related metric information. The IPT will still have the ability to easily modify these documents during the remainder of the development effort. Final approval and formal Configuration Management control will be imposed prior to completion of the validation effort.

4.4 IPT STATEMENT OF WORK IMPLEMENTATION

At the time specified by each APT's master schedule, the IPT will implement the applications software SOW. The process to be followed is based on the level of effort (major or minor) necessary to implement the SOW.

1. A major SOW requires significant up-front work to ensure the timely product delivery. Review panels are strategically used to ensure the process is on track (ref Table 4.4-1). The implementation process is composed of five major components:
    - Requirements Definition and Capture
    - Requirements Analysis and Allocation
    - Software Design
    - Software Production
    - Testing

2. Minor SOWs combine the implementation processes that are required for major SOWs. The ease of minor SOWs allows a single requirement processing phase and the combination of software production and test. Both the ACRP and the RRP are not used in minor SOW processing because of their straight-forward nature. The following processes apply to minor SOWs:
    - Requirements modification
    - Software Analysis and Design
    - Software Production
    - Testing

Table 4.4-1 defines when each of the application implementation processes is required. Minor SOWs have identical processes. Major SOWs differ in the ACRP requirement. The ACRP is not required for external SOWs because RTC Application Software has no choice but to accomplish the task. Internal SOWs require ACRP review to ensure the potential project is workable, given the resources available. The implementation details are described in the next five subsections. Validation and User Acceptance are described in section 4.5.

S/W Process	Statement of Work			
	External		Internal	
	Major Effort	Minor Effort	Major Effort	Minor Effort
App Change Review Panel	Not Required	Not Required	Yes	Not Required
Requirements Capture	Yes	Single	Yes	Single
Requirements Analysis	Yes	Process	Yes	Process
Requirement Review Panel	Yes	Not Required	Yes	Not Required
Software Design	Yes	Single	Yes	Single
Software Production	Yes	Process	Yes	Process
Unit Test	Yes		Yes	
Validation (ref 4.5.1)	Yes	Yes	Yes	Yes
User Acceptance (ref 4.5.2)	Optional	Not Required	*	Not Required

Table 4.4-1 RTC Application Software Process Requirements

*User Acceptance is required for all products delivered during CLCS development. User acceptance applicability is determined by the IPT during the sustaining mode.

4.4.1 Requirements Definition and Capture

To provide a solid foundation for a RTC Application Software CSCI, it is necessary to develop a set of functional requirements that detail the functions, capabilities and expectations of the software.

1. The FRD will be based on the operational characteristics of the vehicle/GSE system. Potential resources for obtaining these characteristics are system Function Designator listings, hardware operating specifications, schematics, LCC/OMRSD documents and GOAL requirements. IPT system knowledge and operational experience are critical to ensuring the necessary functionality is captured.

2. The FRD will be developed using a methodology that describes the requirements in terms of physical world objects.

3. The IPT Lead will remain cognizant of other active IPT activities (via an integrated IPT forum) to be aware of common requirements that could be reused. The Common Applications Library will be searched for potential requirements reuse.

4. The FRD will be formatted per 84K01700, RTC Application Software Documentation Standard.

4.4.2 Requirements Analysis and Allocation

The Requirements Analysis and Allocation phase is a process where the IPT transforms the functional requirements into a software implementation framework. The activities of this phase include:

1. Identification and coordination of all external interfaces between the CSCI under development, other CSCIs and the CLCS system. The IPT coordinates with the APT to work system issues.

2. Generation of “use cases” and scenarios that illustrate the functional requirements and the interaction between objects and events.

3. Identification of potential design/code reuse components (from both the supplier and user viewpoint).

4. Allocation of the functional requirements to the identified CSC/CSUs, based on the above analysis activities. A cross-reference between the FRD and the ODS is developed during this phase.

5. Development of a preliminary ODS to capture the analysis results that apply to design issues and design allocations. The ODS will be formatted per 84K01700, RTC Application Software Documentation Standard. The IPT will assign criticality to CSCs and CSUs.

4.4.3 Software Design

Using the FRD and the preliminary ODS developed during the requirement phases, a more detailed overview design of the CSCI can be developed. The design will adhere to the RTC Application Software Architecture Standard (84K01710). The activities of this phase are:

1. Identification and specification of the classes and objects necessary to implement the functional requirements. The use cases/scenarios are used as the starting point for this activity. RTC Application Software Documentation Standard 84K01700 provides examples of required documentation formats.
    - Class descriptions are enhanced when necessary, by state transition diagrams to help define the class's activities.
    - The classes/objects are scrutinized to identify commonality between objects to support generalization of those objects to as common a base class as possible. Inheritance attributes and methods are also identified during this activity.

2. Identification and specification of the inter-process communications and system interactions. Objects are mapped to the major architectural elements, which assists in the detailing of necessary communication paths.

3. Use cases/scenarios are refined until they contain sufficient information to allow coding.

4. The ODS is updated to capture the design activities and class specifications.

4.4.4 Software Production

During the Software Production phase, the overview design is transformed into software code and associated documentation. Software developers write the code, perform peer reviews and perform preliminary debug testing during this phase.

4.4.4.1 Overview

With the RTC Application Software CSCI functional requirements allocated to individual CSCs/CSUs and the detailed design established, the production phase can begin. The following processes apply:

1. Implementation of each CSC/CSU will be performed using the toolset identified for the CSC/CSU's class of software (e.g., display, object command/control). Reference Section 4.4.4.3.

2. All implementation will adhere to all the applicable programming standards. Reference Section 4.4.4.2.

3. The common starting point for all software development is a search of available reuse resources for components that can be used to accelerate development and enhance maintenance. If a component matches the implementation needs, it will be used. The ODS will be updated to reflect those components used.

4. Requirements traceability will be shown in both CSCs and CSUs (e.g., in a header comment) as well as in the Requirements Traceability Matrix.

5. Prior to exiting this phase of development, each CSC/CSU must be subjected to a peer review to detect any defects of inconsistencies and to ensure reuse resources were used to the maximum extent possible. Detection of problem areas during this phase is crucial to timely RTC Application Software CSCI development.

4.4.4.2 Code Writing

Developed application software code will be compliant with all CLCS Software Programming Standards and Guidelines. Code generated by a COTS tool may be exempt from this requirement; check the standard for exempt requirements.

84K00230	CLCS HCI Style Guide and Standards
84K07500-010	CLCS Programming Standard
84K01720	RTC Application Software Implementation Standard

4.4.4.3 Tools and Languages

All RTC Application Software will be developed using object-oriented methodology. Table 4.4-2 defines the application software tools/languages.

Application component	Tool/Language
Display Monitors (with active data display)	SL-GMS / C++
Display Monitors (without active data display)	SL-GMS / Java / C++
End Item Managers	ControlShell / C++
HCI Overhead Software	Java
CCP Overhead Software	C++
Editor	Xemacs
Documentation Auto Generation	DOC++
Requirements Capture/Definition	DOORS
Configuration Management	Razor
Documentation	MSOffice Products

Table 4.4-2 RTC Application Software Tools and Languages

4.4.4.4 Inspection Process

Prior to exiting the production phase, all new or modified components will be subjected to inspection. During the design phase, each CSC/CSU is identified with a criticality classification. This designation will be used to determine the inspection type required. A review requirement summary is listed in table 4.4-3. Document 84K01730-104 defines the RTC Application Software Peer Inspection Practice.

Change Effort	Type of Component affected		
	Critical	Sensitive	Operational
Major	Peer Inspection	Peer Inspection	Peer Inspection
Minor	Peer Inspection	Walk-through	Walk-through

Table 4.4-3 RTC Application Software Code Review Matrix

4.4.4.4.1 Peer Inspections

For criticality I and II software (critical and sensitive), a peer inspection is performed by a group (two or more) of knowledgeable software production personnel. This inspection is more formal than a walk-through, in that review materials are distributed ahead of time and a scheduled meeting is held. The written inspection results contain both positive and negative findings, along with the reviewers names and inspection date(s). Any rework necessary is coordinated between the author, the inspection leader and the applicable IPT software focal point.

4.4.4.4.2 Walk-Throughs

For Criticality III software (operational support components), a code walk-through will be performed by someone knowledgeable about the code (not the author) designated by the IPT software focal point. The walk-through provides the opportunity for a second look at the code to identify any inconsistencies or coding problems. Any rework necessary is coordinated between the author, the reviewer and the software focal point.

4.4.5 Testing

4.4.5.1 Unit Testing

Unit testing is an informal process performed on software components and programs to check their general functionality and performance in a stand alone mode or with other programs in a debug environment. The purpose of Unit Testing is to debug the software to a degree that it can enter integrated testing with no functional errors.

Unit Testing is the responsibility of the software developer. User participation is encouraged to provide early insight into software functionality and performance. Quality Assurance is not required. Problems encountered during Unit Testing are not required to be documented.

1. Unit Test cases will be developed to perform “black box” or functional testing.

2. Test cases will check both normal and error conditions for each input parameter.

3. All Unit Test cases will be maintained in the Configuration Management common repository for future regression testing.

4. For each code change, the associated Unit Test case will be modified as necessary and then performed successfully before the CSC/CSU can be incorporated into the integrated product. Test outputs should be identical to previous tests for all areas not affected by the change.

5. Test results will be maintained in a common repository to provide for comparisons after follow-on tests.

6. Unit Test cases will be performed on both un-optimized (debug) and optimized (production) versions of the developed source code. Test results shall be identical.

Unit testing is performed on an "as needed" basis throughout the component's life cycle. All newly developed software regardless of criticality will go through Unit Testing. Unit Testing should check the software's functionality including logic and algorithm implementation. Unit Test of reused software should check the interfaces with other software components.

4.4.5.2 Integrated Testing

Integrated Testing is an informal process performed on software to test functionality and performance in an integrated environment with other software components against simulation software and/or a hardware test bed. The purpose of Integrated Testing is to ensure that the software has no known functional and performance problems prior to entering the Validation phase and that the integrated software implementation is acceptable to the software developers. The Integrated Testing environment should approximate the actual real-time operational environment as closely as possible. The complete suite of RTC Application Software which will be used by the subsystem (that is available) should be loaded as well as any other interfacing software. Development or validated simulation software may be used. Integrated Testing may be performed against a hardware test bed (e.g., SAIL, KATS) if deemed appropriate.

Integrated Testing is the responsibility of the software developer. User participation is highly encouraged. Integrated testing offers an excellent opportunity for user familiarization of the software prior to validation testing. Quality Assurance is not required. Problems encountered during Integrated Testing will be documented in the informal tracking system of the Configuration Management Tool. This will provide a history of unit test thoroughness and completeness. This problem tracking is internal to the IPT and does not require special approval for closure.

Integrated Testing will be performed on software when the required components have completed a level of Unit Testing acceptable to the IPT. All newly developed software regardless of criticality is required to go through Integrated Testing.
Integrated Testing should check:

- software functionality
- real-time performance
- user interfaces
- interfaces with reused software components
- interfaces with System Software
- interfaces with other RTC Application Software CSCIs
- interfaces with end item components

4.5

IPT VALIDATION AND USER ACCEPTANCE

The final process in the delivery of applications software is the formal validation and user acceptance performed by systems knowledge personnel. Depending on the change effort and the change criticality, there are different quality support requirements. Validation is a formal process that results in a product that could be used to control hardware. User Acceptance is a method that uses validated software to build confidence before using the product on flight hardware.

4.5.1 Validation

Validation is the formal process performed on software to test functionality and performance in an integrated environment with other software components against simulation software and/or a hardware test bed. Validation ensures the applications software has no known functional or performance problems and that the software implementation is acceptable to the users. Once software has successfully completed Validation, it can be used in an operational environment. The Validation environment should approximate the actual real-time operational environment as close as possible. The complete suite of RTC Application Software which will be used by the subsystem should be loaded along with all interfacing software. Validated simulation software and system software are required to support the RTC Application Software validation phase. Validation may be performed against a hardware test bed (e.g., SAIL, KATS) when deemed necessary by the IPT.

Validation is the responsibility of System Knowledge personnel, with assistance from Software Implementation as required. Quality is required to support validation for changes to all critical components. The IPT and quality assurance will work together to determine the applicability of quality oversight for minor sensitive changes. Quality support requirements are listed in Table 4.5-1

Change Effort	Type of Component affected		
	Critical	Sensitive	Operational
Major	Quality Required	Quality Required	Not required
Minor	Quality Required	Quality Optional	Not required

Table 4.5-1 Validation Quality Support Requirements

Problems encountered during validation will be documented internally in the informal tracking system of the Configuration Management Tool. This will provide a history of unit test thoroughness and completeness. Formal PRACA documentation will be required for problems:

- that will not be corrected prior to release to an operational environment
- that are detected in previously validated software or components.

Documentation of test procedures used and the test results are required (reference Application Software Documentation Standard 84K01700).

Validation will be performed on software when components have completed a level of Unit Testing and Integrated Testing acceptable to the IPT. All software regardless of criticality is required to go through Validation. Validation should test:

- software functionality
- real-time performance
- user interfaces
- interfaces with reused Application Software
- interfaces with System Software
- interfaces with other Applications Software CSCIs
- interfaces with end item components

4.5.2 User Acceptance

User Acceptance is a systematic approach by which the user community gains the experience and builds the confidence in the application (and system) product. User Acceptance culminates in the global use of the application software and underlying system software for vehicle processing. In the development phases of CLCS, user acceptance criteria will be identified by both the IPT and by external sources. External sources include senior Shuttle technical representatives who will formally identify user acceptance requirements to the CLCS project.

User acceptance requirements could include, but is not limited to, such items as:

- Simulated power up/down (via simulation model / SAIL)
- Simulated Cryogenic tanking
- Simulated Launch Countdown (via simulation model / SAIL)
- Actual PAD cryogenic cold flow
- Actual Orbiter power up/down
- Simulated cluster test using the simulation model
- Simulated cluster test using SAIL/KATS

An example user acceptance process:

- Assure the hands on (engineering user) is familiar with the application product
- Use the applications product to perform nominal testing using simulations
- Use the application to perform contingency operations using simulations
- Demonstrate the process against hardware (using Test Preparation Sheet (TPS) or temporary deviation)
- Modify test procedures, as required, for CLCS applications
- Perform a review to verify: familiarization, acceptance testing, and procedure modifications are complete
- “Accept” (i.e., release) the product for daily operational support

The specific software undergoing formal user acceptance should be validated. Quality support is not required during this phase. Any system or application problems encountered during the acceptance process will be documented on formal Problem Reports (PRs) by the person detecting the anomaly. As in all formal reporting systems, the PR must be dispositioned appropriately (i.e., constraint, no constraint and rationale) before application product release.

Validated simulation software is not required during acceptance testing. This allows flexibility in the acceptance phase to exercise far fetched test cases. It also allows for increased user familiarization without incurring the cost of formally validating test cases within the simulation system. Traceability is provided by the configuration management system.

Many of the processes in user acceptance are accomplished during the test / validation phase. For example, nominal and contingency paths are tested extensively during validation. User acceptance is not designed to be “testing”, instead it is a method of building confidence that, those who are required, can operate the applications software. The iterative approach used in the applications process should resolve every error before this phase.

In the sustaining environment, user acceptance is typically implied as the product completes validation. For major changes to an application, the IPT will determine the requirements for user acceptance. The IPT, management and the user community must be comfortable with the applications product before it is released for day to day use.

A. GLOSSARY

ACRP	Applications Concept Review Panel
APT	Application Software Product Team
CCMS	Checkout, Control and Monitoring System
CLCS	Checkout and Launch Control System
COTS	Commercial Off The Shelf
CSC	Computer Software Component
CSCI	Computer Software Configuration Item
CSP	Change Screening Panel
CSU	Computer Software Unit
DDS	Detailed Design Specification
FRD	Functional Requirement Document
GOAL	Ground Operations Aerospace Language
GSE	Ground Support Equipment
HCI	Human Computer Interface
HIM	Hardware Interface Module
IPT	Integrated Product Team
IV&V	Independent Validation and Verification
KATS	Kennedy Avionics Test Set
KSC	Kennedy Space Center
LCC	Launch Commit Criteria
NASA	National Aeronautics and Space Administration
ODS	Overview Design Specification
OMI	Operations and Maintenance Instruction
OMRSD	Operations and Maintenance Requirement Specification Document
PAE	Product Assurance Engineer
PRACA	Problem Reporting and Correction Action
QE	Quality Engineer
ROM	Rough Order of Magnitude
ROR	Responsible Organization Representative
RRP	Requirements Review Panel
RTC	Real Time Control
SAIL	Shuttle Avionics Integration Lab
SDP	Software Development Plan
SDT	System Design Team
SEMP	System Engineering Management Plan
SFOC	Space Flight Operations Contract
SOW	Statement of Work

B. RTC APPLICATION SOFTWARE CSCI

The following table contains the official RTC Application Software CSCI names and descriptions.

CSCI	***Description***
CAS	Common Application Support
APU	Orbiter Auxiliary Power Unit
ARM	Swing Arms
BAP	SRB Auxiliary Power Unit
BHY	SRB Hydraulics
BRS	SRB Range Safety System
CME	Main Engine Controller
COM	Communications
DPS	Data Processing System
ECL	Environment Control and Life Support
ECS	Environmental Control System
EFC	Electronic Flight Controls
EPD	Electrical Power Distribution and Control
FCP	PRSD / FC
GID	Guidance
GLS	Ground Launch Sequencer
HMF	Hypergolic Maintenance Facility
HWS	Hazardous Gas Warning System
HYD	Orbiter Hydraulics
ICE	ET Surface Ice
INS	Instrumentation
INT	Integrated Operations
KUB	KU-Band Radar
LH2	Liquid Hydrogen
LO2	Liquid Oxygen
MEQ	Mechanisms
MPS	Main Propulsion System
MST	Master
NAV	Navigation
OMS	Orbiter Maneuvering System
PLE	Payload Test
SME	Space Shuttle Main Engine
WAT	FIREX Water
CCS	Complex Control System